Amendments to the Specification:

Please replace the paragraph beginning on page 1, line 7 with the following amended paragraph:

The present invention relates generally to an improved data processing system and, in particular, to a method and system for processing ~~JavaServer~~ JAVASERVER pages. Still more particularly, the present invention relates to a method, apparatus, and computer instructions for a configurable ~~JavaServer~~ JAVASERVER pages processing framework to process ~~JavaServer~~ JAVASERVER pages.

Please replace the paragraph beginning on page 1, line 16 with the following amended paragraph:

With increasing use of the Internet, customer demands on information provided by Web pages have become more dynamic. Web developers are not only required to generate presentation of Web pages, but also business logic required to generate dynamic contents, such as, for example, a name and an email address of a user from a server. The demand for a flexible method to build dynamic Web pages contributes to the development of ~~JavaServer~~ JAVASERVER pages, a technology available from Sun Microsystems, Inc.

Please replace the paragraph beginning on page 1, line 26 with the following amended paragraph:

~~JavaServer~~ JAVASERVER pages (JSP) provide Web developers an open, freely available specification that provides the advantages, such as, platform-independence, use of the [[Java]] JAVA programming language, and compatibility with a variety of technologies, such as Web servers, Web browsers, and application servers. A JSP page is a Web page with containing a markup language and additional [[Java]] JAVA code. Examples of markup languages are hypertext markup language (HTML), extensible markup language (XML) and wireless markup language (WML). The [[Java]] JAVA code is embedded within tags. When the JSP page is called, this page is compiled by the JSP engine into a [[Java]] JAVA servlet, which is handled by a servlet engine. The servlet engine loads the servlet class using a class loader and executes the class to create dynamic HTML to be sent to the browser. The next time the same JSP page is called, the JSP engine executes the already-loaded servlet.

Please replace the paragraph beginning on page 2, line 18 with the following amended paragraph:

However, due to increasing complexity of the JSP source file and evolving JSP specification, the use of JSP technology described above has become more costly. The maintenance cost of JSP processors, such as a JSP engine inside an Application server or an integrated development environment (IDE) that authors and tests JSP pages, has also increased. In addition, all JSP pages are required to be validated syntactically for correctness before further processing can be done. With different JSP processors, different requirements are present. For example, an IDE tool may need to present a graphical view of JSP page for editing and a JSP engine may need to generate a [[Java]] <u>JAVA</u> source file from the JSP page.

Please replace the paragraph beginning on page 4, line 2 with the following amended paragraph:

The present invention provides a method, apparatus, and computer instructions for a configurable ~~JavaServer~~ <u>JAVASERVER</u> pages (JSP) processing framework to process JSP. In a preferred embodiment, the processing framework of present invention provides a JSP translator that translates a JSP document or page into a document object model (DOM) object. A JSP processor, such as a service or tool provider, may use the JSP translator to configure custom classes, known as "visitor" classes, to be invoked in a certain sequence, so that custom functions may be performed on the DOM object to return custom results. The configuration of custom classes is stored in XML format using a JSP visitor configuration file. The processing framework of present invention also includes built-in classes and a JSP engine that validates the syntax of the JSP and return any errors if necessary.

Please replace the paragraph beginning on page 6, line 19 with the following amended paragraph:

Figure 11 is a diagram illustrating component interactions for validating a JSP and generating [[Java]] <u>JAVA</u> source file from the JSP in accordance with a preferred embodiment of the present invention.

Please replace the paragraph beginning on page 10, line 13 with the following amended paragraph:

An operating system runs on processor **302** and is used to coordinate and provide control of various components within data processing system **300** in **Figure 3**. The operating system may be a commercially available operating system, such as Windows XP, which is available from Microsoft Corporation. An object oriented programming system such as a [[Java]] JAVA may run in conjunction with the operating system and provide calls to the operating system from [[Java]] JAVA programs or applications executing on data processing system **300**. "[[Java]] JAVA" and "JAVASERVER" are trademarks ~~is a trademark~~ of Sun Microsystems, Inc. Instructions for the operating system, the object-oriented operating system, and applications or programs are located on storage devices, such as hard disk drive **326**, and may be loaded into main memory **304** for execution by processor **302.**

Please replace the paragraph beginning on page 11, line 12 with the following amended paragraph:

The present invention provides a method, apparatus, and computer instructions for a configurable ~~JavaServer~~ JAVASERVER pages (JSP) processing framework to process JSP. In a preferred embodiment, the processing framework of the present invention is configurable to invoke user-supplied classes that meet specific requirements of a JSP processor in a user-defined sequence, in order to gather information about a JSP document or page. The user-supplied classes are known as "visitor" classes.

Please replace the paragraph beginning on page 13, line 5 with the following amended paragraph:

A Web developer may run a JSP processor, such as integrated development environment (IDE) **408,** on client **402** to author or test JSP pages. A Web developer may also run another JSP processor, such as JSP engine **410,** to generate a [[Java]] JAVA source file from the JSP page. Currently, ~~JavaServer~~ JAVASERVER pages (JSP) are stored in a Web server **406**. When a JSP processor, such as IDE **408** or JSP engine **410**, requests a JSP page to be processed, Web server, such as Web server **406** responds by sending the desired JSP page to corresponding processing framework **412** or **414**. Then, processing framework **412** or **414** translates the JSP page into a DOM object using a DOM generator and validates the syntax of the JSP page by invoking built-in classes, such as "ValidateJspVisitor" class. If the syntax is correct, processing framework **412** invokes custom "visitor" classes to present a graphical view of the JSP page for editing, and processing framework **414** invokes custom "visitor" classes to generate [[Java]] <u>JAVA</u> source file from the JSP page. Thus, through the use of processing framework of the present invention, each JSP processor may perform common functions, such as validation of the JSP page, as well as custom functions, such as generating [[Java]] <u>JAVA</u> source file, by invoking custom "visitor" classes.

Please replace the paragraph beginning on page 15, line 7 with the following amended paragraph:

The second primary component of the processing framework is the JSP configuration manager. The JSP configuration manager configures JSP via JSP property groups in web.xml file. A web.xml file is a XML file that defines settings of Web components, such as [[Java]] <u>JAVA</u> servlets and tag library descriptors (TLD). A JSP property group is a group of JSP pages with certain properties, such as page encoding, and expression language evaluation. A given property group applies to JSP that matches a URL pattern that is defined in the group. When a Web module is loaded, JSP property groups are also loaded from the web.xml file.

Please replace the paragraph beginning on page 17, line 1 with the following amended paragraph:

The order of execution is important because one visitor may depend on results of a previous visitor. The number of visits by a visitor within a collection also is important because a visitor may have discrete phases of operation that needs to be called separately. For example, a [[Java]] <u>JAVA</u> code generation visitor has to generate seven discrete sections of a [[Java]] <u>JAVA</u> source file. Each section involves participation of different methods within the class that must be executed in a particular order. In addition, jsp-visitor-collection **606** element includes an id, such as id **620**. This id is the name of a collection, such as "JspTranslation".

Please replace the paragraph beginning on page 18, line 3 with the following amended paragraph:

In addition, JSP visitor configuration file **702** includes a number of jsp-visitor-collections, such as, for example, JspTranslation **722**, DebugJspTranslation **724**, TagFileTranslation **726**, DebugTagFileTranslation **728**, and TagFileDependency **730**. Within each collection, a number of definitions are defined with corresponding execution order and number of visits. For example, JspTranslation **722** includes TagFileDependencyCheck **732** to be executed first, followed by JspValidate **734** and JspGenerate **736**, which is executed seven times in order to generate a [[Java]] <u>JAVA</u> source file. Thus, JSP visitor configuration file enables processing of the JSP document or page to be configurable, so that custom functions may be performed by the JSP processor to satisfy different requirements.

Please replace the paragraph beginning on page 23, line 6 with the following amended paragraph:

In addition to syntax validation, common typical operations, such as collection of included tag files, generation of [[Java]] <u>JAVA</u> source file for a JSP page or document and its tag files, are supported by the processing framework of the present invention.

Please replace the paragraph beginning on page 23, line 11 with the following amended paragraph:

Turning next to **Figure 11**, a diagram illustrating component interactions for validating a JSP and generating [[Java]] <u>JAVA</u> source file from the JSP is depicted in accordance with a preferred embodiment of the present invention. As depicted in **Figure 11**, JSPTranslatorUtil **1102** is a class, which initiates the processing of a JSP document or page. JSPTranslatorUtil **1102** initializes the JSPTranslatorFactory **1104** by invoking the initialize method (call **1120**) with the path of the JSP visitor configuration file, which defines a collection of "visitor" classes.

Please replace the paragraph beginning on page 24, line 5 with the following amended paragraph:

Once all the elements and nodes are validated, JSPTranslator **1106** invokes the visit method (call **1132**) of the next visitor class according to the JSP visitor configuration file, GenerateJSPVisitor **1110**. In the illustrative examples, GenerateJSPVisitor **1110** includes methods that generate seven sections of a [[Java]] JAVA source file for a JSP document or page. The seven sections include import, class, static initializer, init, method init, method, and finally. In this example, two of the seven sections are generated by invoking generateImportSection (call **1134**) and generateInitSection (call **1136**) methods of generateJSPVisitor **1110**.

Please replace the paragraph beginning on page 25, line 3 with the following amended paragraph:

After all dependent tag files are collected, the child nodes of the node that started off this sequence are now processed by invoking the processChildren method (call **1142**). To complete processing of the JSP, tag files collected are validated by invoking visit method (call **1144**) of validateTagFileVisitor **1116**. ValidateTagFileVisitor **1116** invokes visitTagDirectiveStart method (call **1146**) to validate the tag file. Finally, a [[Java]] JAVA source file for the tag file is generated by invoking visit method (call **1148**) of generateTagFileVisitor **1118**, which in turn invokes its generateImportSection method (call **1050**), generateClassSection method (call **1152**), and generateInitSection method (call **1154**). These three methods generate three of the seven sections in a [[Java]] JAVA source file.